Lower Bounds and Complete Problems in Nondeterministic Linear Time and Sublinear Space Complexity Classes

Philippe Chapdelaine and Etienne Grandjean

GREYC, Université de CAEN, Bd Maréchal Juin, 14032 CAEN Cedex, FRANCE
{philippe.chapdelaine,etienne.grandjean}@info.unicaen.fr

Abstract. Proving lower bounds remains the most difficult of tasks in computational complexity theory. In this paper, we show that whereas most natural NP-complete problems belong to NLIN (linear time on nondeterministic RAMs), some of them, typically the planar versions of many NP-complete problems, belong to NTISP(n, n^q), for some $q < 1$, i.e., are recognized by nondeterministic RAMs in linear time and sublinear space. The main results of this paper are the following: as the second author did for NLIN, we

- give exact logical characterizations of nondeterministic polynomial time-space complexity classes;
- derive from them a problem denoted Layered-Constraints(t, s), which is complete in the class NTISP$(n, n^{s/t})$, for all integers t, s, $t \geq s \geq 1$, and
- as a consequence of such a precise result and of some recent separation theorems by [9] using diagonalization, prove time-space lower bounds for this problem, e.g. Layered-Constraints$(3, 2) \notin$ DTISP$(n^{1.618}, n^{o(1)})$.

Key Words: computational complexity, descriptive complexity, finite model theory, complexity lower bounds, time-space complexity.

1 Introduction and discussion

1.1 The difficulty to prove complexity lower bounds

One of the main goals of computational complexity is to prove lower bounds for natural problems. In his Turing Award Lecture [4] twenty years ago, S. Cook noted: "There is no nonlinear time lower bound known on a general purpose computation model for any problem in NP, in particular, for any of the 300 problems listed in [10]". Since 1983, despite of some technical progress (see for example [2, 8, 9, 12, 18, 19, 24]) things have not fundamentally changed and Fortnow [9] wrote in 2000: "Proving lower bounds remains the most difficult of tasks in computational complexity theory. While we expect problems like satisfiability to take time $2^{\Omega(n)}$, we do not know how to prove that non linear-time algorithms exist on random-access Turing machines". In our opinion, the persistent difficulty to prove time lower bounds for natural NP-complete problems is due to the conjunction of two facts.

(i) Nondeterminism makes such problems easy, typically they belong to NLIN, i.e. are recognized in linear time on nondeterministic RAMs, and most of them are even easier, i.e. we conjecture that they are not NLIN-complete.

(ii) Almost nothing is known about the relationships between deterministic time and nondeterministic time.

Let us now develop both arguments (i) and (ii).

1.2 Nondeterminism makes problems easy

In a series of papers [13, 14, 16], the second author showed that many NP-complete problems, including the 21 problems studied in the seminal paper of Karp [20], belong to NLIN, i.e. can be recognized in linear time on nondeterministic RAMs, and [12, 13, 25] (see also [17]) proved that a few of them are NLIN-complete, including the problem Risa (Reduction of Incompletely

Specified Automata: quoted [AL7] in the well-known book [10] of Garey and Johnson). Moreover, [14] and [1] argue that it is unlikely that many NP-complete problems in NLIN such as SAT(the satisfiability problem) and HAMILTON (the Hamilton cycle problem) are NLIN-complete. Further, several authors [21, 22, 26, 29] give convincing arguments that a number of NP-complete problems, including CLIQUE, PARTITION and planar restrictions of NP-complete problems are even easier. E.g., [22] deduced from the Planar Separator Theorem (see [21]) that the MAXIMUM INDEPENDENT SET problem in planar graphs can be computed in deterministic subexponential time $2^{O(n^{1/2})}$, whereas we believe the same result does not hold for many other NP-complete problems including SAT (see [29]). Finally, in the same direction, [16] recently proved that a couple of graph problems, including HAMILTON and CUBIC-SUBGRAPH belong to the class Vertex-NLIN, i.e., are recognized by nondeterministic RAMs in time $O(n)$, where n is the number of vertices of the input graph, which may be much less than the size (number of edges) of the graph.

1.3 Our ignorance of the relationships between deterministic time and nondeterministic time

Whereas most people expect NP-complete problems to be exponential, there are only very modest results that formally prove that nondeterminism gives strictly more power to computation. Interestingly, [24] proved that nondeterministic Turing machines (TM) compute strictly more problems in linear time than deterministic ones, namely $\mathrm{DTIME}_{\mathrm{TM}}(n) \subsetneq \mathrm{NTIME}_{\mathrm{TM}}(n)$. Using this result and the inclusion $\mathrm{NTIME}_{\mathrm{TM}}(n) \subseteq \mathrm{NLIN}$, [12] concludes that RISA (or any other similar NLIN-complete problem via $\mathrm{DTIME}_{\mathrm{TM}}(n)$ reductions) does not belong to $\mathrm{DTIME}_{\mathrm{TM}}(n)$. However, it would be much more significant to obtain the similar but stronger result for deterministic RAMs, namely RISA $\notin$ DLIN, that is equivalent to the conjecture DLIN $\neq$ NLIN. This would be a very strong result since, as argued in [28, 17], the class DLIN exactly formalizes the important and very large class of linear time computable problems.

Despite of our pessimistic arguments (*i-ii*), some progress has been recently made by considering mixed time-space complexity.

1.4 Time-space lower bounds

In recent years, Fortnow and several authors [8, 9, 23, 30] have used a new approach to show that some problems like SAT require a nonminimal amount of time or space. Their techniques inspired by some earlier work of [19] essentially use two arguments sketched below.

1. **A hardness result:** SAT is "complete" for quasi-linear time $O(n(\log n)^{O(1)})$ under reductions that use quasi-linear time and logarithmic space (see [5, 6, 27]);
2. **A separation result proved by diagonalization:** There exist constants a, b such that $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTISP}(n^a, n^b)$ (see [9]), where $\mathrm{DTISP}(T(n), S(n))$ denotes the class of problems computable on deterministic RAMs in time $O(T(n))$ *and* space $O(S(n))$.

From (1-2), Fortnow et al. [9] conclude: for any constants $a' < a, b' < b$: SAT $\notin \mathrm{DTISP}(n^{a'}, n^{b'})$.

Finally, note that another completely different current of research (see e.g. [3, 2, 18])uses combinatorial techniques to prove lower bounds for specific problems. However, to our knowledge such techniques have never been compared to the "hardness-separation" method (1-2) above.

Let us now describe the contribution of this paper.

1.5 Our contribution

In this paper we generalize for mixed time-space complexity classes $\mathrm{NTISP}(n, n^{s/t})$ and $\mathrm{NTISP}^{\sigma}(n^t, n^s)$ (for any signature σ and any integers $t \geq s \geq 1$) the results of [15, 16] about NLIN and similar

time complexity classes $\mathrm{NTIME}^{\sigma}(n^t)$ for RAMs [1]. The organization and the main results of the paper are the following: in Section 2 we show that many significant NP-complete problems belong to some "sublinear" classes $\mathrm{NTISP}(n, n^q)$, $q < 1$. In Section 4 we introduce the logic $\mathrm{ESO}^{\sigma}(r, s)$ and prove the exact characterization $\mathrm{ESO}^{\sigma}(r, s) = \mathrm{NTISP}^{\sigma}(n^{r+s}, n^s)$. In Section 5, we obtain a problem, denoted Layered-Constraints(t, s), that is complete in the class $\mathrm{NTIME}(n, n^{s/t})$ and we deduce lower bounds for this problem in Section 6.

2 Time-linear and space-sublinear classes contain significant problems

One important condition for a complexity class to be pertinent is to contain natural problems. In this section, we show that the class $\mathrm{NTISP}(n, \sqrt{n})$, that trivially contains the Clique problem, contains many planar graph problems and some problems over numbers. Moreover, we show that there also are significant problems in the classes $\mathrm{NTISP}(n, n^{1-\frac{1}{d}})$, for each integer $d \geq 2$.

2.1 The case $\mathrm{NTISP}(n, \sqrt{n})$

The examples given in the case $\mathrm{NTISP}(n, \sqrt{n})$ mainly concern planar graph problems. We first need a separator result for planar graphs that allows a more convenient presentation of the input.

Lemma 1. *[21] Let G be any n-vertex planar graph. The vertices of G can be partitioned into three sets A, B, C such that no edge joins a vertex in A to a vertex in B, neither A nor B contains more than $2n/3$ vertices, nor C contains more than $2\sqrt{2}\sqrt{n}$ vertices. Furthermore, A, B and C can be computed in time $O(n)$.*

Now we can use this lemma recursively, i.e. find a separator set S_1 for A and S_2 for B and so on, until we have subsets of size $O(\sqrt{n})$. In such a way we build a binary tree that represents our graph so that the nodes of the tree are the subsets $S, S_1, S_2, \ldots$ and every edge of the graph joins two vertices in the same node or in two nodes of the same branch (because any (separator) node disconnects its two child subgraphs). We call this tree a *separating tree*.

Lemma 2. *Computing a separating tree T of a planar graph $G = (V, E)$ can be done in time $O(n \log n)$ and space $O(n)$, with $|V| = n$.*

Proof. Consider the following facts.

- The tree is linear in size (i.e., $O(n)$), as its nodes form a partition of the vertices of the graph.
- Each subset of type A or B (i.e., every successive subtree) is of size at most $(2/3)n$, $(2/3)^2 n$, $(2/3)^3 n$, etc. Thus, to reach size $O(\sqrt{n})$, we need $O(\log n)$ steps, so the depth of the tree is $O(\log n)$.
- Now by Lemma 1 a separating step (computing S, A and B) is done in time and space linear in the size of the subgraph involved. Considering that, at any given level of the tree, the union of the subgraphs at the nodes of that level is a subpartition of the whole graph, and so is of size $O(n)$, the whole computation of the separation of all the subgraphs at that level is done in time and space $O(n)$.
- Hence, the whole computation of the tree is done in time $O(n \log n)$ and space $O(n)$.

□

For each planar graph problem, we can give a new version with this representation, for example for the 3-Colourability problem.

[1] $\mathrm{NTISP}^{\sigma}(T(n), S(n))$ denotes the class of σ-problems (i.e. sets of first-order structures of signature σ) recognizable by nondeterministic RAMs in time $O(T(n))$ and space $O(S(n))$ where n is the cardinality of the domain of the input σ-structure. This generalizes the notation of [16].

Problem Separating Tree Planar 3-Colourability

Instance : An undirected planar graph $G = (V, E)$ given in the separating tree representation T.

Question : Is G 3-colourable?

Proposition 1. Separating Tree Planar 3-Colourability *and* Separating Tree Planar Vertex Cover *are in NTISP*$(n, \sqrt{n})$.

In order to prove Proposition 1, we need the following additional result:

Lemma 3. *In each separating tree of a planar graph* $G = (V, E)$*, each branch composed of nodes* $E_0, E_1, \ldots, E_l$ *has* $|E_0| + |E_1| + \ldots + |E_l| = O(\sqrt{n})$ *vertices, where* $|V| = n$.

Proof. Along a given branch, the subtrees at the successive nodes are of size at most n, $2n/3$, $(2/3)^2 n$, etc. So the size of each separator is successively at most $k\sqrt{n}$, $k\sqrt{2n/3}$, $k\sqrt{(2/3)^2 n}$, etc. where $k = 2\sqrt{2}$. Therefore, the size of a branch without its leaf is:

$$\begin{aligned} |E_0| + |E_1| + \ldots + |E_{l-1}| &\leq \sum_{i=0}^{l} k\sqrt{(2/3)^i n} \\ &\leq k\sqrt{n} \sum_{i=0}^{\infty} \sqrt{(2/3)}^i \\ &= O(\sqrt{n}) \end{aligned}$$

Finally, the subset E_l at the leaf is, by definition of the separating tree, of size $O(\sqrt{n})$ so overall the size $|E_0| + \ldots + |E_l|$ of the branch is $O(\sqrt{n})$. □

Proof (of Proposition 1). Consider the first branch of the separating tree. It is of size $O(\sqrt{n})$ by Lemma 3 and so we can guess for it a 3-colour assignment and check that it is correct in time and space each $O(\sqrt{n})$. Now, following a depth-first search algorithm pattern, we can successively forget the assignment at the leaf (we have already checked that it is correct, and no further edge will ever lead to a vertex in this subgraph) and process the next branch. So we can recursively check the 3-colourability of the graph, one branch at a time, while visiting every node at most as often as there are edges leading to it. Therefore at any time, there is never more than one branch in memory, limiting the size to $O(\sqrt{n})$, and the total number of visits to nodes is at most the number of edges, which for planar graphs is $O(n)$.

The proof is similar for Separating Tree Planar Vertex Cover. □

Although there is an $O(n \log n)$ delay to build the tree, which prevents to prove that Plan-3-Col $\in$ NTISP$(n, \sqrt{n})$, this proposition still has a significant consequence concerning an upper bound for this problem.

Proposition 2. *If* NTISP$(n, \sqrt{n}) \subseteq$ DTISP$(T(n), S(n))$*, with* $T(n) \geq n \log n$ *and* $S(n) \geq n$*, then* Plan-3-Col $\in$ DTISP$(T(O(n)), S(O(n)))$.

Moreover, a similar result can be applied to a wide range of problems, namely those linearly equivalent to Plan-3-Col, as stated by the following results.

Lemma 4. *[1]* Plan-3-Col *is linearly equivalent to* Plan-Sat *(i.e., there are* DLIN *[1, 17] reductions both from* Plan-3-Col *to* Plan-Sat *and from* Plan-Sat *to* Plan-3-Col*) and* Plan-Hamilton.

Definition 1. *[1]* LIN-PLAN-LOCAL *is the class of problems linearly reducible to* Plan-Sat.

Corollary 1. *If* $\text{NTISP}(n, \sqrt{n}) \subseteq \text{DTISP}(T(n), S(n))$, *with* $T(n) \geq n \log n$ *and* $S(n) \geq n$, *then* $\text{LIN-PLAN-LOCAL} \subseteq \text{DTISP}(T(O(n)), S(O(n)))$.

This last corollary shows that a result similar to Proposition 2 can be applied to the wide range of problems that are linearly reducible to PLAN-SAT. Note that Corollary 1 is much more precise than the previously known inclusion $\text{LIN-PLAN-LOCAL} \subseteq \text{DTIME}(2^{O(\sqrt{n})})$ [1] (because of the inclusion $\text{NTISP}(n, \sqrt{n}) \subseteq \text{DTIME}(2^{O(\sqrt{n})})$).

Other interesting problems that happen to be in $\text{NTISP}(n, \sqrt{n})$ are the well-known PARTITION [10, ref SP12] and KNAPSACK [10, ref MP9] problems.

Proposition 3. *The problems* PARTITION *and* KNAPSACK *are in* $\text{NTISP}(n, \sqrt{n})$.

Proof. Recall that the PARTITION and KNAPSACK problems are defined as follows.

Problem PARTITION
Instance : A finite set A of integers.
Question : Is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} a = \sum_{a \in A' \setminus A} a$?

Problem KNAPSACK
Instance : A finite set U, for each $u \in U$ a size $s(u) \in \mathbb{N}$ and a value $v(u) \in \mathbb{N}$, and positive integers B and K.
Question : Is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leq B$ and such that $\sum_{u \in U'} v(u) \geq K$?

The idea of the proof that PARTITION belongs to $\text{NTISP}(n, \sqrt{n})$ is based on the one given by Hunt and Stearns in [29] for $\text{DTIME}(2^{O(\sqrt{n})})$. Consider an instance $A = \{a_1, \ldots, a_k\}$ of PARTITION. The size of the input is n, that is the a_i are written in base n, and they occupy n registers. Consider a fixed real d, $0 \leq d < 1$, and compute $n^d - 1$. Consider two empty sets A_1 and B_1, with $S(A_1)$ and $S(B_1)$ the sums of the integers in A_1 and B_1 respectively. Now for each a_i whose size is smaller or equal to $n^d - 1$ registers, nondeterministically put it in A_1 or in B_1 and add its value to $S(A_1)$ or $S(B_1)$. Note that since there are at most n numbers a_i whose size is smaller or equal to $n^d - 1$ registers, then $S(A_1) \leq n \cdot n^{n^d - 1} = n^{n^d}$ (the same for $S(B_1)$) and can be stored in n^d registers. Now we consider all the a_is of size greater than $n^d - 1$ and we nondeterministically partition them into two subsets A_2 and B_2. There clearly are no more than n^{1-d} such numbers and so this is the number of registers needed to keep a record of the partition. Once this is done, we sum the units (in base n) of the numbers in A_2 with the unit digit of $S(A_1)$, and we make sure it is equal modulo n to the sum of the units of the numbers in B_2 and the unit digit of $S(B_1)$. This only takes $O(1)$ registers as we work in base n. We also compute the carry and then we do the same for the tens, the hundreds, etc (in base n). If at every stage, the sums are equal, we have a partition of $\{a_1, \ldots, a_k\}$. Finally, the whole computation uses linear time and space $O(n^d + n^{1-d})$, that is $O(n^{1/2})$ if we set $d = 1/2$.

The proof is similar for the KNAPSACK problem. □

2.2 Classes $\text{NTISP}(n, n^{1-\frac{1}{d}})$

The following parameterized problem shows that each class $\text{NTISP}(n, n^{1-\frac{1}{d}})$, for every integer $d \geq 2$, contains a (quite natural) problem.

Problem d-CONSTRAINT TILING
Instance : An integer $d \geq 2$, d integers $m_1, m_2, \ldots, m_d$ and a d-dimensional $m_1 \times m_2 \times \ldots \times m_d$ grid, with a set of allowed tiles for each hypercube of the grid. Each tile has its faces coloured.

Question : Can we choose for every hypercube of the grid one of its allowed tiles so that two adjacent hypercubes have the same colour on their common face?

Proposition 4. *For every integer $d \geq 2$, the problem* d-CONSTRAINT TILING *is in* $\mathrm{NTISP}(n, n^{1-\frac{1}{d}})$.

Proof. We prove this result for $d = 2$, the general case being an easy generalization. Consider a rectangle consisting of $n \times m$ squares, with each one having a nonempty set of tiles. The size of the input is $t \geq nm$. Suppose that $n \geq m$. Consider the first row of m squares and choose nondeterministically an allowed tile for each square. Now do the same for the second row and check that both row are consistent internally and with each other. Once this is done, the choices made for the first row are of no more use and can be forgotten. The memory space they occupied can be used to hold the tiles chosen for the third row, then checking the consistence with the second row. And so on, until we have in memory the $n-1^{th}$ and n^{th} rows. At any time we only keep 2 rows in memory, each of size m, and the space used is always the same recycled, that is $O(\sqrt{t})$ (recall that $m \leq \sqrt{mn} \leq \sqrt{t}$), and it is easy to see that the whole process takes time $O(t)$. □

3 Computational and logical preliminaries

3.1 NRAMs and time-space complexity classes

The time-space functions studied here being very tight, it is very important to describe precisely the computational model we use, that is the Nondeterministic Random Access Machine (or NRAM) as it was designed by Grandjean and al. in several papers (see for example [15, 28, 16]), with only slight modifications.

An NRAM $\mathcal{R}$ is designed to store an input structure $S = \langle [n], \sigma \rangle$, where $[n] = \{0, 1, \ldots, n-1\}$ and σ is a finite signature containing p function or predicate symbols [2]. It consists of (see Figure 1):

- *input registers*: a register L containing the integer n, and for each σ-symbol f of arity k, and for each tuple $a \in [n]^k$, a register $f[a]$ containing the value of f in a;
- *the working section* composed of $d+1$ *special registers* (called *accumulators*), $A, B_1, \ldots, B_d$, where $d = max_{f \in \sigma}\{arity(f)\}$, and *the main memory* which consists of computation registers $R_0, R_1, \ldots$

Convention 1.
- The input registers are called $Q_j(a)$, where Q_j is the j^{th} symbol of σ, $1 \leq j \leq p$, and $a \in [n]^k$, where p is the number of symbols of σ and k is the arity of Q_j.
- All the input registers are read-only while the computation registers $A, B_1, \ldots, B_d, R_0, \ldots$ are read/write.

The program of the NRAM $\mathcal{R}$ is a sequence of instructions $\mathcal{I}(1), \mathcal{I}(2), \ldots, \mathcal{I}(\lambda)$ of the following types ($1 \leq j \leq p, 1 \leq i \leq d$):

(1) $A := L$	$(8, i)$ $B_i := A$
(2) $A := 0$	$(9, i)$ $R(A) := B_i$
(3) $A := A + 1$	$(10, i)$ *if* $A = B_i$ *then goto* $\mathcal{I}(i_0)$
(4) $A := A - 1$	*else goto* $\mathcal{I}(i_1)$
(5) $guess(A)$	(11) *accept*
$(6, j)$ $A := Q_j(B_1, \ldots, B_k)$	(12) *reject*
(7) $A := R(A)$	

[2] In our notation, we confuse each signature (resp. function or predicate symbol) with its interpretation.

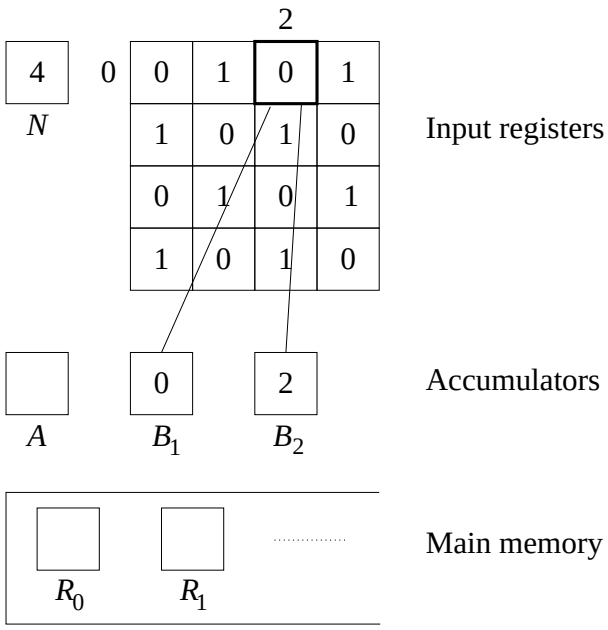

Fig. 1. An NRAM associated with a binary relation

Convention 2. – At the beginning of the computation, all the accumulators and the registers in the main memory contain the value 0.
– *guess*(A) is the non-deterministic instruction of the NRAM; it stores any integer in accumulator A.
– The only *accept* instruction in the program is $\mathcal{I}(\lambda)$, that is the last one.

Remark 1. The access to the main memory is only possible via accumulator A.

Following this definition of our computational model, we can now define the mixed time-space complexity classes we study here:

Definition 2. *Let σ be a signature and $T, S : \mathbb{N} \to \mathbb{N}$ be functions such that $S(n) \leq T(n)$ and $T(n) \geq n$. We call* NTISP$^\sigma(T(n), S(n))$ *the class of problems over σ-structures (or σ-problems) computable on an* NRAM *using time $O(T(n))$ (i.e. that performs $O(T(n))$ instructions) and space $O(S(n))$ (i.e. the registers of the main memory used have adresses $O(S(n))$ and their contents are $O(\max\{n, S(n)\})$), where $[n]$ is the domain of the input σ-structure.*

Notation 1. We will write NTISP$(T(n), S(n))$ as an abbreviation for NTISP$^\sigma(T(n), S(n))$ when σ is a unary $\{f\}$-signature, i.e. f is a unary function symbol. This corresponds to the usual convention since such a structure has size n.

3.2 Formulas and logical classes

We use the standard definitions of logic and finite model theory, see e.g. [7].

Let *succ* be the predefined non-cyclic successor over $[n]$, that is the function

$$\begin{cases} y \mapsto y+1 \text{ if } 0 \leq y < n-1 \\ n-1 \mapsto n-1 \end{cases}$$

For every $\delta \geq 1$, we define the non-cyclic lexicographical successor function over $[n]^\delta$ as the following abbreviation, also denoted $succ(y_1, \ldots, y_\delta)$:

$$succ^{(\delta)}(y_1, \ldots, y_\delta) = \begin{cases} (y_1, \ldots, y_{i-1}, y_i + 1, 0, \ldots, 0) \text{ if } (y_1, \ldots, y_\delta) \text{ is not the} \\ \qquad \text{last } \delta\text{-tuple, i.e. if for some } i = 1, \ldots, \delta, \text{ we} \\ \qquad \text{have } y_j = n - 1 \text{ for each } j > i \text{ and } y_i < n - 1 \\ \\ (n-1, \ldots, n-1) \qquad\qquad \text{otherwise} \end{cases}$$

Definitions 3 and 4 are the cornerstone of the main result, as they show the restrictions we impose on the logic we will use to characterize our complexity classes.

Definition 3. *A first-order quantifier-free formula $\psi(x_1, \ldots, x_s, y_1, \ldots, y_r)$ of signature $\sigma \cup \{0, succ\} \cup \bar{g}$ over $s + r$ variables is called an* (s, r)-restricted formula with input σ *if all the function (resp. relation) symbols of $\bar{g}$ are of arity at most $s + r$, with the following restriction on the arguments of those of arity $s + \delta$ ($\delta \geq 1$): the first s arguments are not restricted and the last δ arguments form a vector of form either $(y_1, \ldots, y_\delta)$ or $succ^{(\delta)}(y_1, \ldots, y_\delta)$.*

Remark 2. Note that if $r = 1$, the last argument of an $(s+1)$-ary function is y or $succ(y)$.

We can now define the classes of logical formulas that will characterize our mixed time-space complexity classes.

Definition 4. *We call* $\mathrm{ESO}^\sigma(s, r)$ *the class of Existential Second Order formulas of the form*

$$\exists \overline{g}\ \forall x_1 \ldots \forall x_s\ \forall y_1 \ldots \forall y_r\ \psi(\overline{x}, \overline{y})$$

where $\overline{g}$ is a set of function or relation symbols of arity at most $s + r$, $y_1, \ldots, y_r$ are called the iteration variables*, and ψ is a quantifier-free (s, r)-restricted formula with input σ and of signature $\sigma \cup \{0, succ\} \cup \{\overline{g}\}$.*

4 Logical characterization of mixed time-space classes

A main result of this paper is the following exact characterization of each time-space complexity class $\mathrm{NTISP}^\sigma(n^t, n^s)$, for all integers $t \geq s \geq 1$, which generalizes the similar characterization of the classes $\mathrm{NTIME}^\sigma(n^t)$ [16]:

Theorem 1. *For all integers $s \geq 1, r \geq 0$ and any signature σ, a σ-problem $\mathcal{P}$ is in* $\mathrm{NTISP}^\sigma(n^{s+r}, n^s)$ *iff there exists a formula ϕ in* $\mathrm{ESO}^\sigma(s, r)$ *that characterizes $\mathcal{P}$, i.e. such that for every σ-structure $\langle [n], \sigma \rangle$ of domain $[n]$:*

$$\langle [n], \sigma \rangle \in \mathcal{P} \text{ iff } \langle [n], \sigma, succ, 0 \rangle \models \phi \tag{1}$$

For the sake of simplicity and without loss of generality, we will restrict ourselves to signatures $\sigma = \{f\}$ containing only one function symbol f, of any arity d. That means the program of the NRAM will contain instructions $(6, j)$ of the unique following form:

$$(6) \quad A := f(B_1, \ldots, B_d)$$

Also we will prove the theorem only for the case $r = s = 1$ the general case being just an easy generalization of this particular one. Note that we will use the linear order $<$ over domain $[n]$ as it is definable in $\mathrm{ESO}^\emptyset(1, 1)$ (see [11] or [16, Corollary 2.1]).

First, if there exists a formula ϕ in $\mathrm{ESO}^\sigma(1, 1)$ such that the equivalence above holds for every σ-structure, then it is easy to see that $\mathcal{P}$ is in $\mathrm{NTISP}^\sigma(n^2, n)$.

Let $\mathcal{P}$ be a σ-problem. Suppose that there exists an $\mathrm{ESO}^\sigma(1, 1)$ formula ϕ such that the equivalence (1) of Theorem 1 holds for every σ-structure $\langle [n], \sigma \rangle$. An NRAM $\mathcal{R}$ can check ϕ in the following way:

- $\mathcal{R}$ first *guesses* and stores the p_1 unary ESO functions $g_i : [n] \to [n]$, and the $2 \times (p - p_1)$ unary restrictions of the binary ESO functions $g_i : [n] \times [n] \to [n]$ by setting $y = 0$ and $y = 1$. All this can be done in linear time $O(n)$ and by using a linear number of registers. We can then check the formula for all x and for $y = 0$ (remember the form of a $(1, 1)$-restricted formula).
- We now have the following loop: for $y = 1$ to $y = n - 1$, replace in the registers the values of the binary functions for $y - 1$ by the values for y, and then *guess* the values for $y + 1$ and store them in the registers just freed. Check whether the formula holds for all x and for the current y. This is done also in linear time; the space used is the same as in the first step, so it is still linear.

We have n such iterations (including the one for $y = 0$), each of time $O(n)$, so the time used overall is quadratic. The space used is always the same, that is linear. So we have $\mathcal{P} \in \mathrm{NTISP}^{\sigma}(n^2, n)$.

Now let us see how we can describe a problem in $\mathrm{NTISP}^{\sigma}(n^2, n)$ with a formula in $\mathrm{ESO}^{\sigma}(1, 1)$. A problem $\mathcal{P}$ is in $\mathrm{NTISP}^{\sigma}(n^2, n)$ iff it is recognized by an NRAM $\mathcal{R}$ that works in time at most cn^2 and space cn, and uses numbers at most cn, for some fixed integer c. We can also suppose that if our NRAM works in time less than cn^2, then the final configuration is repeated until instant cn^2 so that $\mathcal{R}$ works in time exactly cn^2 and the instants of the computation can be labelled $0, 1, 2, \ldots, cn^2 - 1$.

Our goal is to describe the computation of $\mathcal{R}$ with a logical formula. As we cannot describe the content of every register at any time (this would require a size $\Theta(\text{time} \times \text{space}) = \Theta(n^3)$, i.e. some ternary function on the domain $[n]$), we only encode what may change: the current instruction index, the contents of accumulators $A, B_1, \ldots, B_d$, and the content of the register pointed to by A. We want a logical formula over the domain $[cn]$, so we will code the instants of the computation $0, 1, \ldots, cn^2 - 1$ with ordered pairs (t, T), $0 \leq t < cn$, $0 \leq T < n$, so that (t, T) encodes the instant $t + T \cdot cn$ of a computation of $\mathcal{R}$. Let us introduce the following functions:

- $I(t, T)$ denotes the index of the instruction performed at instant (t, T).
- $A(t, T)$ denotes the content of accumulator A at instant (t, T).
- $B_i(t, T)$, $1 \leq i \leq d$, denotes the content of accumulator B_i at instant (t, T).
- $R_A(t, T)$ denotes the content of register $R(A)$ (ie. the register pointed to by A) *at* instant (t, T).
- $R'_A(t, T)$ denotes the content of the same register $R(A)$ *after* step (t, T).

Let us mention two things. Firstly, the encoding of the time naturally divides the time-space diagram of the computation of $\mathcal{R}$ into n blocks of cn instants. Secondly, all those functions are binary and should respect the conditions of Definitions 3 and 4 for the logic $\mathrm{ESO}^{\sigma}(1, 1)$. This compels us to consider only two blocks at once : the current one referred at by T, and the previous one $(T - 1)$. So, T is our *unique iteration variable*. Now, remember that at any instant t of block T, we must be able to know the contents of accumulators A, $B_1, \ldots, B_d$ and of the register pointed to by A. The successive contents of the accumulators are completely described by the above functions A and $B_1, \ldots, B_d$, so there is no problem for them. In contrast, the contents of the computation registers are accessible only through the function R_A, which must hold the right value at any time, considering the fact that the register pointed to by A at instant (t, T) may be distinct from the one at the previous instant. Moreover, if a specific register is not accessed in two consecutive blocks, the restriction imposed to the iteration variable T seems to prevent the recovery of its content. So the problem is to be able to get the value that was stored in any register the last time it was accessed, be it in the current block or in any other block before. The idea to overcome this difficulty is to resume, at the beginning of each block T $(0 \leq T < n)$, the content of any computation register of address x $(0 \leq x < cn)$ with a binary

function $R(x,T)$. More precisely, $R(x,T)$ will code the content of register $R(x)$ at the instant $(0,T)$, that is the instant $cn \cdot T$ of the computation of $\mathcal{R}$. We are now ready to give the formulas that describe the computation of $\mathcal{R}$. First, the initial conditions are described by formula ϕ_1:

$$\phi_1 \equiv I(0,0) = 1 \ \wedge \ A(0,0) = 0 \ \wedge \ B_1(0,0) = 0 \wedge \ldots \wedge B_d(0,0) = 0$$

Functions I, A, B_i, $1 \leq i \leq d$, and R'_A can be easily defined by recurrence from I, A, B_i and R_A by $\mathrm{ESO}^\sigma(1,1)$ formulas ϕ_I, ϕ_A, ϕ_{B_i} and $\phi_{R'_A}$ respectively.

We use the following conventions:

- We use two different successor functions. The first one is the one described above ($succ^{(1)}$) and will be applied to T. The second one is the successor function over $[cn]$ and it will be applied to t. Both are denoted by $succ$ since they have roughly the same behaviour. Moreover, we use the abbreviation $succ(t,T)$ for:

$$succ(t,T) = \begin{cases} (succ(t),T) & \text{if } t < cn-1 \\ (0,succ(T)) & \text{if } t = cn-1 \text{ and } T < n-1 \\ (cn-1,n-1) & \text{otherwise} \end{cases}$$

- The input function $f : [n]^d \to [n]$ is padded by the function

$$\begin{aligned} F : \quad & [cn]^d \to [cn] \\ & (x_1,\ldots,x_d) \mapsto \begin{cases} f(x_1,\ldots,x_d) \text{ if } x_i < n \text{ for all } i \leq d, \\ 0 \text{ otherwise} \end{cases} \end{aligned}$$

  This function will be the only one in the input signature of the $(1,1)$-restricted formula, and so will not be restricted in its arguments.
- We use the function $pred$ (non-cyclic predecessor) easily definable in $\mathrm{ESO}^\sigma(1,1)$.
- $I(t,T) = (i)$ (for $i = 1,\ldots,12$) is an abbreviation for the disjunction $\bigvee_{j \in S_i} I(t,T) = j$ where $S_i \subseteq \{1,2,\ldots,\lambda\}$ denotes the set of indices of instructions of type (i) in the program of $\mathcal{R}$.

We have the following case definitions:

$$I(succ(t,T)) = \begin{cases} i_0 & \text{if } I(t,T) = (10,i) \text{ and } A(t,T) = B_i(t,T) \\ i_1 & \text{if } I(t,T) = (10,i) \text{ and } A(t,T) \neq B_i(t,T) \\ I(t,T) & \text{if } I(t,T) = (11) \text{ or } (12) \\ succ(I(t,T)) & \text{otherwise} \end{cases}$$

$$A(succ(t,T)) = \begin{cases} n & \text{if } I(t,T) = (1) \\ 0 & \text{if } I(t,T) = (2) \\ succ(A(t,T)) & \text{if } I(t,T) = (3) \\ pred(A(t,T)) & \text{if } I(t,T) = (4) \\ G(t,T) & \text{if } I(t,T) = (5) \\ F(B_1(t,T),\ldots,B_d(t,T)) & \text{if } I(t,T) = (6) \\ R_A(t,T) & \text{if } I(t,T) = (7) \\ A(t,T) & \text{otherwise} \end{cases}$$

where the non-deterministic feature of the instruction $guess(A)$ is given by the ESO function $G(t,T)$.

$$B_i(succ(t,T)) = \begin{cases} A(t,T) & \text{if } I(t,T) = (8,i) \\ B_i(t,T) & \text{otherwise} \end{cases}$$

$$R'_A(succ(t,T)) = \begin{cases} B_i(t,T) \text{ if } I(t,T) = (9,i) \\ R_A(t,T) \text{ otherwise} \end{cases}$$

These case definitions of I, A, B_i and R'_A can be easily transformed into first-order formulas ϕ_I, ϕ_A, ϕ_{B_i} and $\phi_{R'_A}$ respectively, of the form $\forall T < n\ \forall t\ \psi(t,T)$ where ψ is some $(1,1)$-restricted formula.

So the computation of $\mathcal{R}$ between two successive instants will be expressed by ϕ_2:

$$\phi_2 \equiv \phi_I\ \wedge\ \phi_A\ \wedge\ \phi_{B_1} \wedge \ldots \wedge \phi_{B_d}\ \wedge\ \phi_{R'_A}$$

Now there remains to describe functions R_A and R, which is a much more tricky task. For this purpose, we introduce, as in [15], the binary function $N(x,T) = (N_1(x,T), N_0(x,T))$ (more precisely two binary ESO function symbols N_0, N_1) which represents, in each block T, the lexicographical numbering of the ordered pairs $(A(t,T),t)$ (see Figure 2 for an example on a given block T):

$$\begin{aligned} \phi_N \ \equiv & \forall T < n\ \forall t\ \exists x \\ & N_1(x,T) = A(t,T)\ \wedge\ N_0(x,T) = t \\ & \wedge\ x \neq cn-1\ \rightarrow\ N(x,T) <_{lex} N(succ(x),T) \end{aligned}$$

where $(i,j) <_{lex} (i',j')$ abbreviates $i < i' \vee (i = i' \wedge j < j')$.

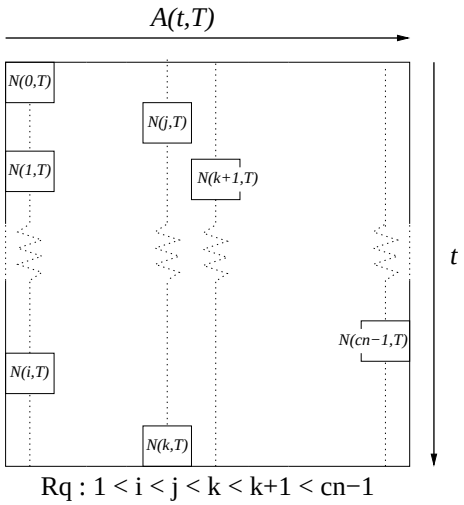

Fig. 2. $N(x,T)$

Notice that the first two conjuncts of ϕ_N express that, for every $T < n$, the mapping $x \mapsto N(x,T)$ is a surjection, and hence is a bijection from the set $[cn]$ to the set of equal cardinality $\{(A(t,T),t) : t \in [cn]\}$.

Remark 3. We use the non-cyclic successor functions over $[cn]$ and over $[n]$. Both are denoted *succ* as they have the same behaviour.

The binary function R that allows to represent the content $R(x,T)$ of register $R(x)$ at instant $(0,T)$ is described by formula ϕ_R:

$$
\begin{aligned}
\phi_R \equiv\ & \forall T < n-1\ \forall x\ \exists t\ \exists z\ \exists t'\ \exists u \\
& \{T = 0 \rightarrow R(x,T) = 0\} \\
& \left\{ \begin{array}{l}
\left\{ \begin{array}{l}
A(t,T) = x\ \wedge\ N(z,T) = (A(t,T),t) \\
\wedge \left[\begin{array}{ll}
z = cn-1\ \vee & \left(\begin{array}{l} z \neq cn-1 \\ \wedge\, N(succ(z),T) = (A(t',T),t') \\ \wedge\, A(t,T) \neq A(t',T) \end{array} \right)
\end{array} \right] \\
\wedge\, R(x,succ(T)) = R'_A(t,T)
\end{array} \right\} \\
\\
\bigvee \left\{ \begin{array}{l} N(0,T) = (A(t,T),t) \\ \wedge\, [A(t,T) > x\ \wedge\ R(x,succ(T)) = R(x,T)] \end{array} \right\} \\
\\
\bigvee \left\{ \begin{array}{l} N(cn-1,T) = (A(t,T),t) \\ \wedge\, [A(t,T) < x\ \wedge\ R(x,succ(T)) = R(x,T)] \end{array} \right\} \\
\\
\bigvee \left\{ \begin{array}{l} A(t,T) < x < A(u,T) \wedge\ (A(t,T),t) = N(z,T) \\ \wedge\, (A(u,T),u) = N(succ(z),T) \\ \wedge\, R(x,succ(T)) = R(x,T) \end{array} \right\}
\end{array} \right\}
\end{aligned}
$$

Remark 4. The first line of the matrix of ϕ_R (first conjunct) describes the behaviour of R in the first block (labelled 0). In the big second part (second conjunct), the first disjunct corresponds to the case when register $R(x)$ is accessed in block T (in particular, the formula in brackets $[\ldots]$ combined with the condition $A(t,T) = x$ expresses that (t,T) is the last instant in block T when $A(t,T) = x$). The other three disjuncts correspond to the three cases when register $R(x)$ it is not accessed in block T. See Figure 3 for more details.

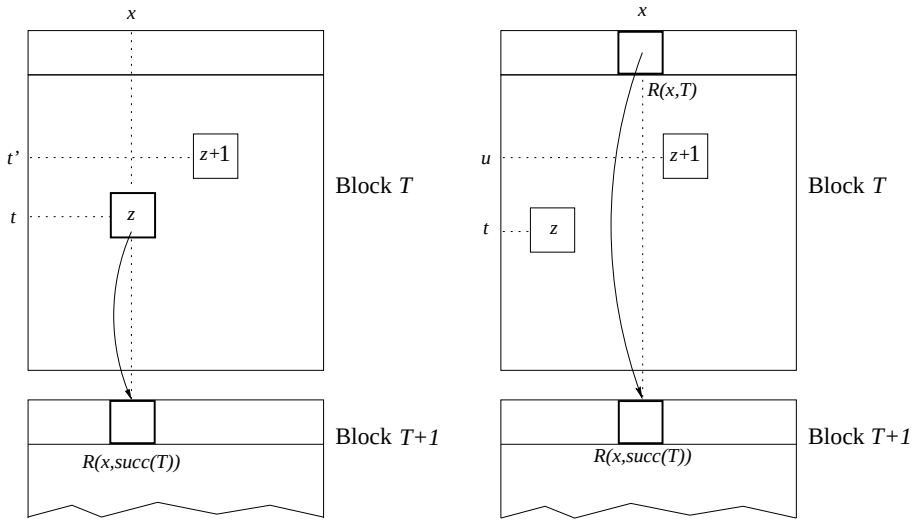

Fig. 3. $R(x, succ(T))$

By using functions R and R'_A, function R_A (which describes the right content of the register pointed to by accumulator A) is easily defined by formula ϕ_{R_A} (see Figure 4):

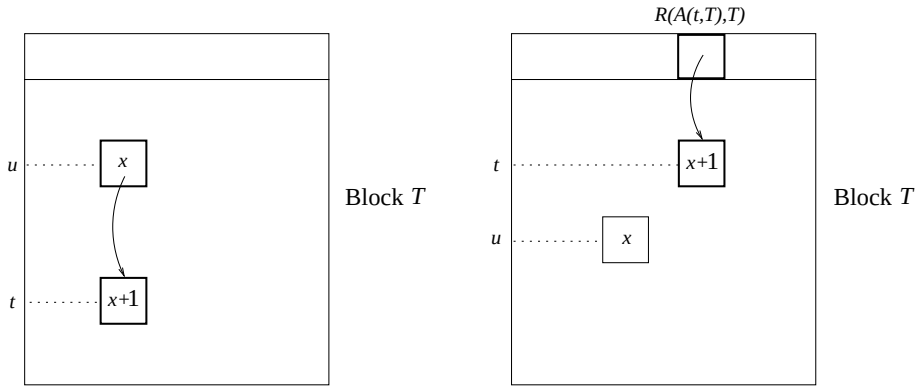

Fig. 4. $R_A(t,T)$

$$\begin{aligned}\phi_{R_A} \equiv\ & \forall T < n\ \forall t\ \exists x\ \exists u \\ & \{(A(t,T),t) = N(0,T)\ \wedge\ R_A(t,T) = R(A(t,T),T)\} \\ & \bigvee \left\{ \begin{array}{l} \quad (A(t,T),t) = N(succ(x),T)\ \wedge\ (A(u,T),u) = N(x,T) \\ \wedge\ (A(t,T) = A(u,T)\ \rightarrow\ R_A(t,T) = R'_A(u,T)) \\ \wedge\ (A(t,T) \neq A(u,T)\ \rightarrow\ R_A(t,T) = R(A(t,T),T)) \end{array} \right\}\end{aligned}$$

So we control the contents of all the registers via formula ϕ_3:

$$\phi_3 \equiv \phi_N\ \wedge\ \phi_R\ \wedge\ \phi_{R_A}$$

The fact that $\mathcal{R}$ reaches the *accept* instruction, that is $\mathcal{I}(\lambda)$, is ensured by formula ϕ_4:

$$\phi_4 \equiv I(cn-1, n-1) = \lambda$$

Finally, the computation of $\mathcal{R}$ is exactly described by formula ϕ over domain $[cn]$:

$$\phi \equiv \exists I, A, B_1, \ldots, B_d, R_A, R'_A, G, N\ \phi_1\ \wedge\ \phi_2\ \wedge\ \phi_3\ \wedge\ \phi_4$$

and it is easy to transform ϕ into a prenex Skolemized $\mathrm{ESO}^{\sigma}(1,1)$ formula.

We have described the computation of the NRAM $\mathcal{R}$ on an input f by an $\mathrm{ESO}(1,1)$ formula for the structure $\langle [cn], F\rangle$, that means over domain $[cn]$; now let us see how to get a formula for the input structure $\langle [n], f\rangle$, i.e. over domain $[n]$. The idea is to code an element $x \in [cn]$ by an ordered pair of elements $(x_0, x_1) \in [n] \times [c]$. According to this idea, every binary function $g : [cn] \times [n] \rightarrow [cn]$ will be coded by $2c$ functions ${g_0}^{(i)} : [n] \times [n] \rightarrow [n]$ and ${g_1}^{(i)} : [n] \times [n] \rightarrow [c]$, for $0 \leq i < c$, defined as follows:

$$\begin{aligned} {g_0}^{(i)} &: [n] \times [n] \rightarrow [n] \\ & \quad (x,y) \mapsto g(in + x, y) \bmod n \end{aligned}$$

$$\begin{aligned} {g_1}^{(i)} &: [n] \times [n] \rightarrow [c] \\ & \quad (x,y) \mapsto \left\lfloor \frac{g(in+x,y)}{n} \right\rfloor \end{aligned}$$

So, for every $g : [cn] \times [n] \rightarrow [cn]$, every $y \in [n]$ and every $(x_0, x_1) \in [n] \times [c]$, we have $g(nx_1 + x_0, y) = n \cdot {g_1}^{(x_1)}(x_0, y) + {g_0}^{(x_1)}(x_0, y)$. We notice that the iteration variable, that is $y = T$, is not modified and hence our binary functions respect our restricted logic (see Definitions 3 and 4). The process is similar for any unary function $g : [cn] \rightarrow [cn]$. The details of the proof are left to the reader [15]. □

Now, the computation of $\mathcal{R}$ is exactly described by formula ϕ over domain $[n]$.

5 Completeness results for some logical problems

Before presenting our problems, along with some form of completeness for linear time and sub-linear space complexity classes, we need some technical tools.

5.1 A technical result

It will be convenient to encode any set of unary $\{f\}$-structures $\mathcal{P} \in \text{NTISP}(m^{t/d}, m^{s/d})$ - where s, t, d are fixed integers such that $t \geq s \geq 1$ and $t \geq d \geq 1$ and m denotes the size of the unary input structure $\langle [m], f \rangle$ - into a set $\mathcal{P}^{code}$ of d-ary structures.

Remark 5. We use in that notation the letter m instead of n to make the following encoding easier.

For fixed numbers t, s with $t \geq s \geq 1$ and any signature σ, remember that $\text{NTISP}^{\sigma}(n^t, n^s)$ denotes the class of problems over σ-structures $\langle [n], \sigma \rangle$ recognizable by an NRAM that uses computation register contents $O(n^s)$ and works in time $O(n^t)$ and space $O(n^s)$.

Definition 5. *For any unary $\{f\}$-structure $S = \langle [m], f \rangle, f : [m] \to [m]$, let $code(S) = \langle [n], g \rangle$ denote the structure of signature $\sigma_d = \{g_0, \ldots, g_{d-1}\}$, where every g_i is of arity d, defined by $(1-2)$:*

1. *$n - 1 = \lceil m^{1/d} \rceil$, i.e. $(n-2)^d < m \leq (n-1)^d$;*
2. *$g : [n]^d \to [n]^d$ is such that $g = (g_0, \ldots, g_{d-1})$, where $g_i : [n]^d \to [n]$, and*
   - 2.1. *if $f(a) = b$ for any $a, b < m$, then $g_i(a_0, a_1, \ldots, a_{d-1}) = b_i$ where a_i, b_i are the respective i^{th} digits of a, b in base $n-1$, that means $a = \sum_{i<d} a_i (n-1)^i$ and $b = \sum_{i<d} b_i (n-1)^i$ with $a_i, b_i < n-1$, and*
   - 2.2. *$g_i(a_0, a_1, \ldots, a_{d-1}) = n - 1$ if $(a_0, a_1, \ldots, a_{d-1}) \in [n]^d$ is not the list of $(n-1)$-digits of any integer smaller than m.*

Let $\mathcal{P}^{code} = \{code(S) : S \in \mathcal{P}\}$.

The following remarks are essential.

Remark 6. S and $code(S)$ have about the same size, i.e. $size(code(S)) = \Theta(n^d) = \Theta(m) = \Theta(size(S))$.

Remark 7. The correspondence $S \mapsto code(S)$ is one-one and is easily computable as its converse is because if $S' = \langle [n], g \rangle = code(S)$ for $S = \langle [m], f \rangle$ then we have

$$m = \sharp \left\{ (a_0, a_1, \ldots, a_{d-1}) \in [n-1]^d : g_0(a_0, a_1, \ldots, a_{d-1}) < n - 1 \right\} \tag{2}$$

Here is our technical lemma:

Lemma 5. *For any fixed numbers t, s such that $t \geq s \geq 1$ and $t \geq d \geq 1$, $\mathcal{P} \in \text{NTISP}(m^{t/d}, m^{s/d})$ implies $\mathcal{P}^{code} \in \text{NTISP}^{\sigma_d}(n^t, n^s)$.*

Proof. Under the hypothesis, let us give an

Algorithm for recognizing the problem $\mathcal{P}^{code}$
Input : a d-ary structure $S' = \langle [n], g \rangle$.
begin
– compute m with the expression (2) of Remark 7;

– check that all the conditions (1-2) of Definition 5 are satisfied, that means $S' = code(S)$ for some (unique) unary structure $S = \langle [m], f \rangle$ with $n - 1 = \lceil m^{1/d} \rceil$;
– on the input S, simulate the running of the NRAM that recognizes $\mathcal{P}$ in time $O(m^{t/d}) = O(n^t)$ and space $O(m^{s/d}) = O(n^s)$. (Note that this NRAM only uses register contents $O(m) = O(n^d)$.)

end

This proves $\mathcal{P}^{code} \in \text{NTISP}^{\sigma_d}(n^t, n^s)$. □

5.2 Completeness result

Let us now present our logical problems, denoted Layered-Constraints(t, s), where t, s are fixed integers, $t \geq s \geq 1$.

Definition 6. *An* $[n]$*-formula* F *of signature* σ *is a quantifier-free first-order* σ*-formula where the variables are replaced by integers in* $[n]$*. Let* $length(F)$ *denote the number of occurrences of integers,* σ*-symbols, equalities, and connectives in* F*.*

Example: $g(h(1,0),2) = h(3,1)$ is an atomic [4]-formula of signature $\{g, h\}$ and of length 9.

Problem Layered-Constraints(t, s)
Instance : – an integer n;
– a non-empty set $\mathcal{OP}$ of t-ary operators $[n]^t \to [n]$, each explicitly given by its table;
– a conjunction of $[n]$-formulas $F = F_1 \wedge \ldots \wedge F_l$, each F_i of size at most n^s and of signature ν_i, where $\nu_i \cap \nu_j \subseteq \mathcal{OP}$ if $|i - j| > 1$ $(*)$.
Question : Is F satisfiable?

Convention 3. To satisfy automatically condition $(*)$, it is natural to partition the overall signature of F as $\tau_0 \dot\cup \tau_1 \dot\cup \ldots \dot\cup \tau_l \dot\cup \mathcal{OP}$, where $\dot\cup$ denotes the disjoint union, with $\nu_i = \tau_{i-1} \cup \tau_i$, $\tau_0 = \emptyset$. The j^{th} symbol of τ_i is denoted f^i_j.

Remark 8. The size of the input is $m \geq n^t$;

Proposition 5. Layered-Constraints(t, s) *is in* NTISP$(m, m^{s/t})$.

Proof. Let $(n, \mathcal{OP}, F_1 \wedge \ldots \wedge F_l)$ be an instance of Layered-Constraints(t, s). First, we consider formula F_1 and prove that it is coherent with $\mathcal{OP}$. Remember that we use a nondeterministic RAM. Each part of the formula is checked in the following way.

– If it is of the form $F'_i \vee F'_j$, with F'_i and F'_j subformulas of F_1, then we nondeterministically choose F'_i *or* F'_j and check its coherence.
– If it is of the form $F'_i \wedge F'_j$, with F'_i and F'_j subformulas of F_1, then we check if F'_i *and* F'_j are coherent.
– If it is of the form $f(\alpha) = \beta$, with f a symbol of ν_1, $\beta \in [n]$ and $\alpha \in [n]^k$ where k is the arity of f, then we set $f(\alpha) = \beta$ and store it in the memory.
– If it is of the form $f(\alpha) = g(\beta)$, with f, g symbols of ν_1 and $\alpha \in [n]^k, \beta \in [n]^m$ where k is the arity of f and m the arity of g, then we nondeterministically choose an interpretation for f in α and give the same value to g in β.
– The same applies if we have compositions of the form $f(g_1(\ldots), \ldots, g_k(\ldots))$, whatever the arity of the functions involved.
– Whenever an element of $\mathcal{OP}$ appears, we take the interpretation given in the input.

Note that, as the size of F_1 is at most n^s, we don't need more than n^s registers to store the values we need. (Note also that a symbol is identified with the signature in which it appears, so it is easy to see if a formula uses a forbidden symbol.) Once this is done for formula F_1, we sort all the values we have given to the functions, there are at most $|F_1|$ so it can be done in time and space $O(|F_1|)$ (see [14]), and we check that if there are twice (or more) the same symbol in the same value, then the same interpretation is given every time. All this is done in time $O(|F_1|)$ and space $O(n^s)$. We do the same thing for F_2, but as there may be common symbols in both formulas F_1 and F_2, we check that they were given the same value (as both lists are sorted, it can be done at the same time as the check for repeated occurrences). This again is done in time $O(|F_1| + |F_2|)$ and space $O(n^s)$. Now remember that symbols in F_1 no longer occur in the other formulas F_i for $i > 2$, so we can forget their interpretations as they can no longer bring incoherence. The memory thus freed will be used to store the values that appear in F_3, and so on until we check the coherence of F_l.

Overall, the same memory space $O(n^s) = O(m^{s/t})$ is always recycled and the time needed is $O(\sum_{1 \leq i \leq l} |F_i|) = O(m)$, where m is the size of the input. So LAYERED-CONSTRAINTS$(t, s) \in$ NTISP$(m, m^{s/t})$. □

The following theorem essentially expresses the completeness of the problem LAYERED-CONSTRAINTS(t, s) in the class NTISP$(m, m^{s/t})$.

Theorem 2. *For all integers t, s, $t \geq s \geq 1$, and all functions T, S:*

(i) LAYERED-CONSTRAINTS$(t, s) \in$ DTISP$(T(O(m)), S(O(m)))$ *if and only if* NTISP$(m, m^{s/t}) \subseteq$ DTISP$(T(O(m)), S(O(m)))$*;*

(ii) LAYERED-CONSTRAINTS$(t, s) \in$ co-NTISP$(T(O(m)), S(O(m)))$ *if and only if* NTISP$(m, m^{s/t}) \subseteq$ co-NTISP$(T(O(m)), S(O(m)))$.

Proof (Sketch of). Proposition 5 yields the *if* implication. We will prove the *only if* part of this theorem for the case $s = 1, t = 2$, the general case being an easy generalization of this particular one.

Let $\mathcal{P}$ be a problem in NTISP$(m, m^{1/2})$. Then by Lemma 5, $\mathcal{P}^{code} \in$ NTISP$^{\sigma_2}(n^2, n)$, with $\sigma_2 = \{g_0, g_1\}$, and by Theorem 1 there exists a formula φ in ESO$^{\sigma_2}(1, 1)$ such that for each integer $n > 0$ and each σ_2-structure $\langle [n], g_0, g_1 \rangle$ we have

$$\langle [n], g_0, g_1 \rangle \in \mathcal{P}^{code} \text{ iff } \langle [n], g_0, g_1, succ, 0 \rangle \models \varphi \tag{3}$$

Let $\varphi \equiv \exists \bar{f}\ \forall x\ \forall y\ \psi(x, y)$ where y is the iteration variable and ψ is quantifier-free. Without loss of generality, assume that $\bar{f}$ consists of function symbols f_i of arity 2. The idea consists in unfolding the first-order part of φ on the domain $[n]$. This gives the equivalent $[n]$-formula

$$\bigwedge_{b<n} \left[\bigwedge_{a<n} \psi(a, b) \right]$$

which is also equivalent to the conjunction $\bigwedge_{b<n} F_b$ where $F_b = \left[\bigwedge_{a<n} \psi_b(a)\right]$ and $\psi_b(a)$ denotes the formula $\psi(a, b)$ in which each term $f_i(a, b)$ (resp. $f_i(a, succ(b))$), for $f_i \in \bar{f}$, is replaced by $f_i^b(a)$ (resp. $f_i^{b+1}(a)$). In other words, each ESO function symbol $f_i \in \bar{f}$ (of arity 2) is replaced by n function symbols f_i^b $(b < n)$ of arity 1. By construction, we have $\langle [n], g_0, g_1, succ, 0 \rangle \models \varphi$ iff the $[n]$-formula $F \equiv \bigwedge_{b<n} F_b$ is coherent with the tables of functions g_0 and g_1. Note that, by construction, if $|b - b'| > 1$, we have $signature(F_b) \cap signature(F_{b'}) \subseteq \{g_0, g_1\}$.

There remains a technical problem. (n, σ_2, F) is not exactly an instance of the problem LAYERED-CONSTRAINTS$(2, 1)$ since $length(F_b) = kn = N$ for each $b < n$, where $k = length(\psi)$. Therefore, we "linearly pad" our instance into an "equivalent" instance over $[N]$ by completing

the tables of g_0 and g_1 on the domain $[N]$, under the names $g_0^{(N)}$ and $g_1^{(N)}$ respectively: we add the values $g_i^{(N)}(a_0, a_1) = (0, 0)$ whenever a_0 or a_1 belongs to $[N] \setminus [n]$. We obtain an instance $(N, \mathcal{OP}, F)$ of Layered-Constraints$(2, 1)$ with $N = kn$ and $\mathcal{OP} = \left\{g_0^{(N)}, g_1^{(N)}\right\}$ such that (by (3))

$$\langle [n], g_0, g_1 \rangle \in \mathcal{P}^{code} \text{ iff } (N, \mathcal{OP}, F) \in \text{Layered-Constraints}(2, 1)$$

Let us recapitulate the properties of our reduction of any problem $\mathcal{P}$ of NTISP$(m, m^{1/2})$ to Layered-Constraints$(2, 1)$.

- **It is correct:**
  For any m and any input unary structure $S = \langle [m], f \rangle$, if $S' = code(S) = \langle [n], g_0, g_1 \rangle$:

  $$S \in \mathcal{P} \text{ iff } S' \in \mathcal{P}^{code} \text{ iff } (N, \mathcal{OP}, F) \in \text{Layered-Constraints}(2, 1)$$

- **It is linear-sized:**

  $$length(F) = \Theta(N^2) = \Theta(n^2) = \Theta(size(S')) = \Theta(size(S))$$

  and similarly for $size(\mathcal{OP})$.
- **It yields the "only if" implication of the Theorem 2(i):**
  Let $\mathcal{R}$ be a (deterministic) RAM that decides
  Layered-Constraints$(2, 1)$ in time $O(T(O(m)))$ and space $O(S(O(m)))$. The RAM $\mathcal{R}'$ with the following program decides the problem $\mathcal{P}$.
  Input : a unary $\{f\}$-structure $S = \langle [m], f \rangle$.
  begin
  - Compute $n = \lceil m^{1/2} \rceil$ and $N = kn$.
  - Simulate running $\mathcal{R}$ on input $(N, \mathcal{OP}, F)$ without storing it nor $S' = code(S)$. Whenever $\mathcal{R}$ needs to read an operator value $g_i^{(N)}(a_0, a_1) = b_i$ ($i = 0$ or $i = 1$), such that, e.g., $a_0 < n - 1, a_1 < n - 1$ and $a_0 + a_1(n - 1) < m$, $\mathcal{R}'$ reads the value $v = f(a_0 + a_1(n-1))$ in its input S and compute $b_0 = v \bmod (n-1)$ or $b_1 = \lfloor v/(n-1) \rfloor$. Whenever $\mathcal{R}$ needs to read a symbol in F, the easy structure of $F \equiv F_0 \wedge F_1 \wedge \ldots \wedge F_{n-1}$ allows $\mathcal{R}'$ to compute that symbol in constant time and constant space.

  end
  Since the reduction $S \mapsto (N, \mathcal{OP}, F)$ is linear-sized, $\mathcal{R}'$ decides $\mathcal{P}$ within time $O(T(O(m)))$ and space $O(S(O(m)))$ as required. The proof of part (ii) of Theorem 2 is similar. □

6 Corollaries: time-space lower bounds

The completeness results for the problems Layered-Constraints(t, s) obtained in the previous section yield some lower bounds for those problems because of several separation results proved by Fortnow et al. [9] that we reformulate as follows.

Theorem 3 (Fortnow-Melkebeek 2000: See Corollary 4.8, Corollary 3.23, Corollary 3.22, respectively).

- NTISP$(m, m^{0.619}) \not\subseteq$ DTISP$(m^{1.618}, m^{o(1)})$
- NTISP$(m, m^{3/4}) \not\subseteq$ co-NTISP$(m^{1.4}, m^{o(1)})$
- NTISP$(m, m^{4/5}) \not\subseteq$ co-NTISP$(m^{5/4}, m^{1/10})$

Corollary 2.
- Layered-Constraints$(8, 5) \in$ NTISP$(m, m^{5/8}) \setminus$ DTISP$(m^{1.618}, m^{o(1)})$
- Layered-Constraints$(3, 2) \in$ NTISP$(m, m^{2/3}) \setminus$ DTISP$(m^{1.618}, m^{o(1)})$
- Layered-Constraints$(4, 3) \in$ NTISP$(m, m^{3/4}) \setminus$ co-NTISP$(m^{1.4}, m^{o(1)})$
- Layered-Constraints$(5, 4) \in$ NTISP$(m, m^{4/5}) \setminus$ co-NTISP$(m^{5/4}, m^{1/10})$

7 Conclusion and open problems

It is well-known that most natural NP-complete problems are in NLIN and that many of them, e.g. SAT, are complete in nondeterministic quasi-linear time. Moreover, in this paper we have shown that significant NP-complete problems, mainly many problems over planar graphs and some problems over numbers, belong to NTISP(n, n^q), for some $q < 1$, and specifically to NTISP$(n, n^{1/2})$. This improves the known upper bound DTIME$(2^{O(n^{1/2})})$ for those problems. Thanks to a logical description of nondeterministic *polynomial* time-space classes we have exhibited, for any integers s, t, $t \geq s \geq 1$, a problem, denoted LAYERED-CONSTRAINTS(t, s), that is complete in NTISP$(n, n^{s/t})$ via linear reductions. This is a very precise and nontrivial result. The main open challenge would be to discover "more natural" complete problems in such classes, mainly in NTISP$(n, n^{1/2})$, even via quasi-linear reductions. Further, in order to prove some complexity lower bound, such a result should be completed by some separation result for NTISP$(n, n^{1/2})$ that would be similar to those obtained by [9] for any class NTISP(n, n^q), with q greater than 0.619 (the golden ratio).

References

[1] R. Barbanchon and E. Grandjean. Local problems, planar local problems and linear time. In *Computer Science Logic*, volume 2471 of *LNCS*, pages 397–411. Springer, 2002.

[2] Paul Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM Journal on Computing*, 20(2):270–277, 1991.

[3] Alan Borodin and Stephen A. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM Journal on Computing*, 11(2):287–297, 1982.

[4] Stephen A. Cook. An overview of computational complexity. *Communications of the ACM*, 26(6):400–408, June 1983.

[5] Stephen A. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26(5):269–270, 1988.

[6] A. K. Dewdney. Linear time transformations between combinatorial problems. *Intern. J. Comp. Math.*, 11:91–110, 1982.

[7] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, 1995.

[8] Lance Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60:337–353, 2000.

[9] Lance Fortnow and Dieter van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 2–13, 2000.

[10] M. R. Garey and D. S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[11] Etienne Grandjean. First-order spectra with one variable. *Journal of Computer and System Sciences*, 40(2):136–153, 1990.

[12] Etienne Grandjean. A nontrivial lower bound for an NP problem on automata. *SIAM Journal on Computing*, 19(3):438–451, 1990.

[13] Etienne Grandjean. Linear time algorithms and NP-complete problems. *SIAM Journal on Computing*, 23(3):573–597, 1994.

[14] Etienne Grandjean. Sorting, linear time and the satisfiability problem. *Annals of Mathematics and Artificial Intelligence*, 16:183–236, 1996.

[15] Etienne Grandjean and Frédéric Olive. Monadic logical definability of nondeterministic linear time. *Computational Complexity*, 7:54–97, 1998.

[16] Etienne Grandjean and Frédéric Olive. Graph properties checkable in linear time in the number of vertices. *Journal of Computer and System Sciences*, 2003. to appear.

[17] Etienne Grandjean and Thomas Schwentick. Machine-independent characterizations and complete problems for deterministic linear time. *SIAM Journal on Computing*, 32(1):196–230, 2002.

[18] Yuri Gurevich and Saharon Shelah. Nondeterministic linear-time tasks may require substantially nonlinear deterministic time in the case of sublinear work space. *Journal of the ACM*, 37(3):674–687, 1990.

[19] R. Kannan. Towards separating nondeterminism from determinism. *Mathematical Systems Theory*, 17:29–45, 1984.

[20] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[21] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, (36):177–189, 1979.

[22] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.

[23] Richard J. Lipton and Anastasios Viglas. On the complexity of SAT. In *IEEE Symposium on Foundations of Computer Science*, pages 459–464, 1999.

[24] Wolfgang J. Paul, Nicholas Pippenger, Endre Szemerédi, and William T. Trotter. On determinism versus non-determinism and related problems. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 429–438, 1983.

[25] S. Ranaivoson. Nontrivial lower bounds for some NP-complete problems on directed graphs. In *CSL '90*, volume 533 of *Lecture Notes in Computer Science*, pages 318–339, 1991.

[26] J. M. Robson. Subexponential algorithms for some NP-complete problems. Manuscript, 1985.

[27] Claus-Peter Schnorr. Satisfiability is quasilinear complete in NQL. *Journal of the ACM*, 25(1):136–145, 1978.

[28] Thomas Schwentick. Algebraic and logical characterizations of deterministic linear time classes. In *Proc. 14th Symposium on Theoretical Aspects of Computer Science STACS 97*, pages 463–474, 1997.

[29] R. E. Stearns and H. B. Hunt. Power indices and easier hard problems. *Mathematical Systems Theory*, 23(4):209–225, 1990.

[30] Iannis Tourlakis. Time-space lower bounds for SAT on uniform and non-uniform machines. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 22–33, 2000.